Dynamically Reconfigurable Hybrid Cache: An Energy-Efficient Last-Level Cache Design

Yu-Ting Chen, Jason Cong, Hui Huang, Bin Liu, Chunyue Liu, Miodrag Potkonjak, and Glenn Reinman
Computer Science Department, University of California, Los Angeles
Los Angeles, CA 90095, USA
Email: {ytchen, cong, huihuang, bliu, liucy, miodrag, reinman}@cs.ucla.edu

***Abstract*—The recent development of non-volatile memory (NVM), such as spin-torque transfer magnetoresistive RAM (STT-RAM) and phase-change RAM (PRAM), with the advantage of low leakage and high density, provides an energy-efficient alternative to traditional SRAM in cache systems. We propose a novel reconfigurable hybrid cache architecture (*RHC*), in which NVM is incorporated in the last-level cache together with SRAM. RHC can be reconfigured by powering on/off SRAM/NVM arrays in a way-based manner. In this work, we discuss both the architecture and circuit design issues for RHC. Furthermore, we provide hardware-based mechanisms to dynamically reconfigure RHC on-the-fly based on the cache demand. Experimental results on a wide range of benchmarks show that the proposed RHC achieves an average 63%, 48% and 25% energy saving over non-reconfigurable SRAM-based cache, non-reconfigurable hybrid cache, and reconfigurable SRAM-based cache, while maintaining the system performance (at most 4% performance overhead).**

I. Introduction

The traditional SRAM-based on-chip cache has become a bottleneck for energy-efficient design due to its high leakage power. Designers have turned their attention towards emerging non-volatile memories, such as the spin-torque transfer magnetoresistive RAM (STT-RAM) and phase change RAM (PRAM), to build future memory systems. Power, performance, and density characteristics of the new memory technologies differ dramatically compared to SRAM, and thus they enlarge the landscape of memory design.

Table I shows a brief comparison of SRAM, STT-RAM, and PRAM technologies. The exact access time and dynamic power depend on the cache size and the peripheral circuit implementation. In sum, SRAM suffers from the high leakage and low density while providing great endurance; STT-RAM and PRAM provides high density and low leakage at the cost of weak endurance. Moreover, STT-RAM outperforms PRAM in terms of the endurance, access time, and dynamic power, while PRAM has higher density.

TABLE I
Comparison among SRAM, STT-RAM, PRAM.

	SRAM	STT-RAM	PRAM
Density	1X	4X	16X
Read time	Very fast	Fast	Slow
Write time	Very fast	Slow	Very slow
Read power	Low	Low	Medium
Write power	Low	High	High
Leak. power	High	Low	Low
Endurance	10^{16}	4×10^{12}[1]	10^{9}

With desirable characteristics on leakage power and density, NVMs have been explored as an efficient alternative for either SRAM or DRAM in memory systems [2][3][4][5][6]. As can be seen in Table I, compared to PRAM, STT-RAM has higher endurance (10^9 versus 4×10^{12} write cycles) [7][1]. Based on the write cycles, we use a similar endurance model proposed in [8] to calculate the lifetime of PRAM and STT-RAM in an on-chip hybrid cache which consists of 1MB SRAM and 3MB NVM. Table II demonstrates the lifetime of the hybrid cache for three write-intensive workloads selected from medical imaging domain [9] and PARSEC [10]. For a PRAM-based hybrid cache, the lifetime is limited, ranging from 4.70 to 196.12 days; but the STT-RAM-based hybrid cache can last for more than tens of years. Thus, STT-RAM is more suitable for on-chip last-level cache [2][3][4][5] design while PRAM is promising as an alternative for DRAM in the main memory design [6]. Therefore, in this paper, we will focus on a hybrid cache architecture with STT-RAM as the NVM.

TABLE II
Energy of 4MB RHC and 2MB SRAM-based cache

Workloads	registration	segmentation	fluidanimate
PRAM (days)	4.70	196.12	39.33
STT-RAM (years)	12.88	537.32	107.76

A common problem in existing hybrid cache designs [2][3] is the lack of adaptation to varied workloads. Previous studies show that different applications may exhibit different characteristics [11]. For example, if the targeted application accesses a 10MB working set in a streaming fashion, then a fixed hybrid cache design consisting of a 2MB SRAM and 8MB NVM (as discussed in [3]) may become inefficient in terms of both performance and energy compared to a 2MB SRAM-only design. In the 2MB SRAM-only design, all data blocks are put into the SRAM to achieve fast access. However, in the 2MB SRAM with 8MB NVM design, the data blocks are distributed in both the SRAM and NVM regions, while most blocks are located in the NVM region. The cache miss rates are the same for the two architectures due to the streaming access pattern, but the performance degrades because of the longer access latency of NVM. The energy consumption in the hybrid design is larger since it consumes more leakage due to longer runtime and additional leakage from NVM arrays. Also, the greater dynamic write energy on NVM increases the total energy consumption. Therefore, if we can provide configurability on the hybrid cache design, it can be reconfigured to accommodate varied workloads. In this example, the hybrid cache can be reconfigured into 2MB SRAM to achieve the best performance and energy efficiency.

In this paper, we propose a novel reconfigurable hybrid cache design (*RHC*). Our design explores the use of NVM cache to replace the conventional all-SRAM design in the last-level cache to efficiently reduce leakage energy. The proposed RHC design supports reconfigurable SRAM/NVM size, with the capability of powering on/off SRAM and NVM arrays in a way-based manner for better accommodation of memory requirements from different workloads. Hardware-based mechanisms are proposed to detect the cache demand for dynamic reconfiguration. On average, RHC significantly saves 64%, 46% and 28% energy over a non-reconfigurable SRAM cache, a non-reconfigurable hybrid cache and a reconfigurable SRAM cache, respectively, while maintaining the system performance (at most only 4% performance overhead).

II. Related Work

Because of the desirable characteristics on leakage power and density, NVMs have been intensively investigated recently as an

978-3-9810801-8-6/DATE12/©2012 EDAA

efficient alternative for either SRAM in the on-chip caches or DRAM in main memory [2][3][4][5][6]. In [2], STT-RAM and PRAM are used to implement the lower-level cache. Two types of hybrid cache architectures are evaluated – inter-level and intra-level, in which NVMs are utilized either as the entire L3 cache or the slow-accessed region in L2 cache. In [3], 3D stacking STT-RAM is used to build a hybrid cache system with SRAM. However, none of the prior works have considered dynamic powering on/off of SRAM and NVM arrays to adapt to varied workloads.

Dynamically reconfigurable caches are investigated for pure SRAM caches to either reduce the energy consumption via power gating [12][13][14][15][16], or provide dynamic flexible support of software-managed memories to the core through a cache line control bit [17][18]. The key to these approaches is the dynamic assessment of runtime cache pressure. In [12], researchers use a single miss counter to measure the demand of an instruction cache to perform reconfiguration. Missing tags or victim tags are used in [13][14][18] to assess the cache pressure. When a cache miss occurs, the tag of the victim block will overwrite the LRU tag of the same set in victim tags and will be marked as the MRU victim tag. If there is a cache miss and victim tag hit, this indicates that a potential hit would occur if the requested block were held in the cache. The authors in [15] use a time-based counter for each cache block, which will be reset once there is a hit to that block. Once the time while the counter maintaining non-zero status exceeds a given decaying period, the block will be turned off to save leakage. However, none of the existing dynamic reconfiguration schemes have considered hybrid memory technologies.

To the best of our knowledge, this paper is the first work to explore the dynamic cache reconfiguration for hybrid memory technologies in order to reduce the cache energy consumption.

There are also works investigating endurance reduction for the NVMs. In [6][19], wear-leveling techniques are proposed for a PRAM-based memory system to enhance the lifetime. Recent work in [1] uses periodic set-remapping to distribute the writes among sets in a STT-RAM cache. Another study migrates the write-intensive cache blocks to other cache lines in the same/different cache set or in the SRAM to reduce the average write frequency of the STT-RAM (or PRAM) cache lines [8]. These works are orthogonal and complementary to our proposed reconfigurable hybrid cache designs.

III. RECONFIGURABLE HYBRID CACHE DESIGN

In this section we discuss the RHC design in the following way. First, we present the architecture and circuit design of the RHC with disparate SRAM and NVM technologies. Next, the reconfigurability support for the RHC is discussed. Finally, we propose hardware-based schemes for dynamic reconfiguration.

A. Hybrid Cache Architecture

Figure 1 shows an overall structure of RHC. In RHC the data array is partitioned into SRAM and NVM at a cache-way granularity. One concern of RHC design is that the access latency of a NVM cell is longer than that of SRAM [2]. In a simple hybrid cache design where the tag and data arrays of each cache way are implemented either with all SRAM cells or NVM cells, the cache critical path will always be dominated by the longer access latency to the NVM cache ways. To overcome this, RHC is designed in the following way. First, the accesses to the tag array and data array are done sequentially, (i.e. the data array will be accessed after the tag array). Such a serialized tag/data array access has already been widely adopted in a modern low-level large-scale cache for energy reduction. Second, the RHC tag array is fully implemented with SRAM cells. In RHC, each tag entry contains only four bytes, including the tag, coherence state bits, and the dirty bits, etc., while each cache block in the data array contains 64 bytes. Hence, the SRAM-based RHC tag array will not create a large energy overhead.

The circuit design of RHC with STT-RAM as the NVM is as follows. First, an STT-RAM cell has a bitline (BL) and a source-line (SL) for its operation. This is similar to the bitlines (BL, BLB) used in SRAM. Therefore, the organization of an STT-RAM data array is almost the same as an SRAM data array. Second, the sense amplifiers need to be modified due to the single-ended bitlines [4]. According to a recent implementation [20] of an STT-RAM array, the reference voltage is 1.2V, which is close to an SRAM-based design. Therefore, additional power-supply pins to support the read/write accesses of the STT-RAM array may not be required.

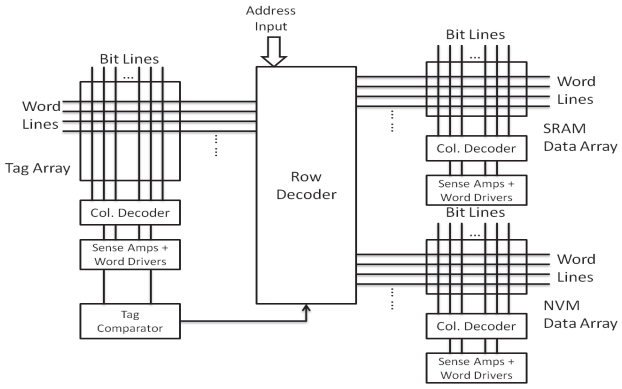


Fig. 1. Reconfigurable hybrid cache (RHC) design

B. RHC Reconfiguration Design

The reconfiguration in RHC is realized by powering on/off SRAM and NVM arrays arbitrarily in a way-based manner. From an architectural point of view, the reconfiguration mechanism in RHC is similar to the existing way-based reconfigurable SRAM cache [21]. Data accesses will not be directed to a disabled cache way, thus those ways in the data array dissipate no dynamic power. Note that the replacement decision logic within the cache controller must ensure that no data will be allocated to a disabled cache way.

In Figure 2 we illustrate the power-gating design adopted in RHC. A centralized power management unit (*PMU*) is introduced to send sleep/wakeup signals to power on/off each SRAM or NVM way. The power-gating circuits of each way in SRAM tag/data arrays are implemented with NMOS sleep transistors to minimize the leakage. In this design the stacking effect of three NMOS transistors from the bitline to GND substantially reduces leakage [12]. Note that in RHC the SRAM cells in the same cache way will be connected to a shared virtual GND while the virtual GNDs among different cache ways are disconnected. This can ensure that the behaviors of cache ways that are powered-on will not be influenced by the powering-off process in other ways.

For the peripheral circuits, such as the row decoder, column decoders, word drivers, and sense amplifiers, we use PMOS sleep transistors to implement the power-gating design; this can provide better performance of the peripheral circuits in the active mode [16]. Since the NVM cell itself consumes little leakage, we do not introduce extra power-gating circuits for the cells of NVM data arrays. To power on/off a NVM cache way, PMU will send a sleep/wakeup control signal to the peripheral circuits of the corresponding NVM way. The design complexity of the PMU is highly related to the adopted wakeup scheme. In this work we assume a daisy-chain wakeup scheme for each cache way [22]. For the PMU, we use Synopsys SAED 90nm technology, which is the most advanced process technology available, to obtain the energy and delay numbers. An RTL-level description of PMU is synthesized by

Synopsys Design Compiler^{TM}. The dynamic energy is 0.0135pJ for one reconfiguration, while the leakage power is 1.0378uW. The delay of the PMU is 0.28ns. The overhead of the PMU is small and thus can be neglected.

The overhead of the reconfiguration will be classified in the following two categories. First, when a cache way is disabled, the dirty blocks in that cache way need to be written back to lower-level memory. This will introduce both performance and energy overhead. Second, from a circuit-level perspective, the power-up process also involves extra energy consumption. The reason for this is that the accumulated charge during the standby mode in SRAM cells should be discharged.

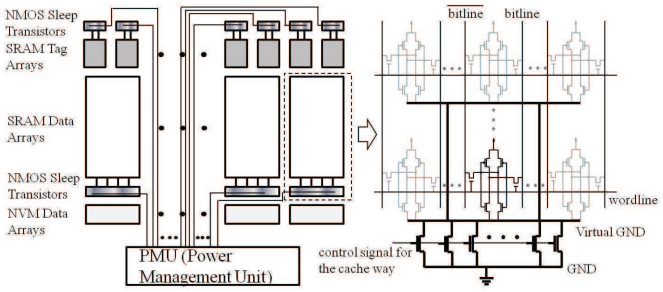


Fig. 2. Power-gating design for RHC

C. Dynamic Reconfiguration

In this section, we propose two hardware schemes to utilize the reconfigurability provided by RHC. The main idea is to detect the cache demand dynamically and reconfigure the RHC in a way-based manner to satisfy the demand. In the meantime, the powered-off cache ways can provide energy savings in leakage.

1) Way-Based Decay Scheme: The reconfiguration scheme includes two dynamic decisions: (1) when to power off a cache way and (2) when to power on a cache way. To power off a cache way, we utilize the *cache decay* idea [15], and introduce the novel *way-based decay counters*, as shown in Figure 3. The main idea of cache decay is to power off a cache block which is not accessed for a long time period to save leakage. This time period is called the *decay interval*. Cache decay is implemented by a local 2-bit saturating counter for each block with a global cycle counter. The local counter is incremented when the global counter exceeds a certain number of clock cycles, which is used to model the decay interval. The local counter is reset to zero when there is an access on this block. Cache decay is initially used to provide a self-guided block-based power-on/off mechanism [15]. However, it is not feasible for PMU to arbitrate reconfiguration at the block-based granularity due to circuit design complexity. Therefore, we use the way-based decay counter to measure the number of decay blocks in that cache way during a time period. The way-based decay counter is incremented by one when any local 2-bit counter in that cache way saturates. Similarly, when a local 2-bit counter is reset to zero, the corresponding way-based decay counter is decreased by one. If the value of a way-based decay counter exceeds a given threshold in a given time period, such as 90% (used in this work) of the blocks in that way, the whole cache way will be powered off due to the low cache demand.

To detect the demand for powering on more cache ways, we keep the whole tag array powered-on to record potential hits if those blocks are in RHC. The potential hit counter is increased by one when a hit occurs on a tag entry whose corresponding data block is powered off (referred to as the *victim tags* [14][18]). The victim tags reuse the same tag array, which does not create extra storage overhead. The replacement policy for the victim tags follows LRU policy. When the value of potential hit counter is greater than a threshold, a cache way is powered on to reduce cache misses. We denote this powering-on threshold as TH_{on}. Note that the power-on/off decision is made for every one-million cycles in this work. This time period is called the *reconfiguration period*. Both the way-based decay counters and the potential hit counter are reset to zero after the decision is made.

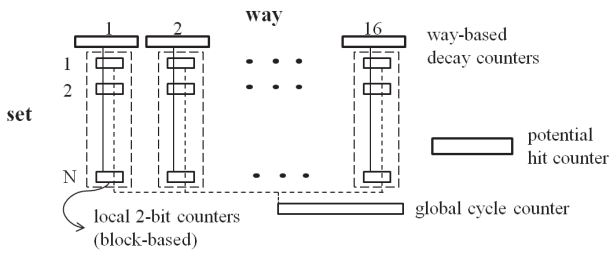


Fig. 3. Counters for dynamic reconfiguration

2) Independent Potential Hit Counters Scheme: In this section, we provide an improved strategy to for dynamic reconfiguration. In the way-based decay scheme, a large number of cache ways can be powered off simultaneously since each cache way is controlled independently. However, we observe that this aggressive powering-off scheme may result in significant performance degradation due to the increase of L2 cache misses especially when the decay interval is small, such as one million cycles. Another potential problem is that a single decay interval cannot accurately capture the varied decay intervals of all cache blocks, which also makes the way-based decay counter ineffective. When the decay interval is too large, such as 100 millions cycles, most of the blocks are accessed once during that interval. The powering-off decisions are seldom made and thus the energy reduction is limited.

The improved scheme takes both the hybrid nature of RHC and the aggressive powering-off issue into consideration. Considering the hybrid nature of RHC, it is beneficial to measure the cache demand for the SRAM and STT-RAM arrays independently to better accommodate the cache demand. Therefore, we use two potential hit counters to measure the cache demand of the SRAM and STT-RAM arrays independently.

The powering-off strategy is different from that of the way-based decay scheme. Here, we use the same potential hit counter to make the powering-off decision. We introduce another powering-off threshold (TH_{off}). When the value of the potential hit counter is less than or equal to TH_{off}, a cache way can be powered off. Based on the strategy, only one cache way can be powered off at a time period, and this greatly reduces the chance of cache thrashing. However, according to our observation, this strategy still generates considerable cache misses. To mitigate the aggressive powering-off strategy, we further restrict the powering-off condition. When the value of the potential hit counter reaches TH_{off}, we cannot power off a cache way immediately. A cache way can only be powered off after ten consecutive reconfiguration periods (ten-million cycles) have passed. Note that TH_{on} is set to 50 and TH_{off} is set to 0 for both SRAM and STT-RAM arrays for evaluation.

Furthermore, we consider the endurance of RHC when making decisions of reconfiguration. We achieve this by randomly selecting the cache way from all possible candidates. For example, when the decision is to power off a cache way, we will randomly pick the victim from all powered-on cache ways.

IV. Evaluation Methodology

A. Performance and Energy Models

We evaluate the proposed RHC design on a simulation platform built upon Simics [23] with GEMS [24]. Table III shows the parameters used in our model. The value *K* represents the number of

cache ways that are powered on in a specific L2 cache configuration, which also equals the amount of "active" cache associativity. Notice that the configuration of the processor core, L1 caches, and main memory remains the same through all simulations.

TABLE III
SIMULATION PARAMETERS

	single-thread workload	multi-thread workload
#Core	1	4
Core	Sun UltraSPARC-III Cu processor core, 4GHz	
L1 Cache	32KB per core for I/D caches 4-way, 64-byte block, 1-cycle latency	
L2 Cache	RHC: 1MB SRAM + 3MB STT-RAM SRAM-based: 2MB K-way (K$\leq$16), 64-byte block	
L2 Cache Access Lat.	SRAM: 10 cycles STT-RAM read/write: 11/30 cycles	
Main Memory	4GB, 320-cycle access latency	

For the energy of the memory technologies, we use the ITRS 32nm process model. The SRAM and STT-RAM energy/latency numbers used in our simulations are obtained from CACTI 6.5 [25] and the data scaled from [4], respectively. The energy numbers of a 4MB RHC and 2MB SRAM-based cache are listed in Table IV, where *Active* and *Standby* correspond to the power-on and power-off state. The standby leakage is estimated according to the ratio of active/standby leakage presented in [12]. This can be achieved through a careful power-gating design.

TABLE IV
ENERGY OF 4MB RHC AND 2MB SRAM-BASED CACHE

L2 Cache Design	Tech.	Dyn. energy per acc. (nJ)	Active leak.(mW)	Standby leak.(mW)
4MB RHC	SRAM	0.137	431.30	14.38
	STT-RAM	Read: 0.278 Write: 0.765	116.92	3.897
2MB SRAM		0.288	711.29	23.71

B. Benchmarks

Our testbenchs consist of 16 benchmark applications, which have been carefully chosen to represent memory intensive algorithms in the fields of data processing, massive communication, scientific computation and medical applications. The applications include seven memory-intensive applications from SPEC2006 [26], four applications from PARSEC [10], and five applications from the medical imaging domain [9].

TABLE V
WORKLOADS

Benchmark	Applications
SPEC2006	bzip2, mcf, soplex, libquantum, h264ref, lbm, astar
PARSEC(simmedium)	blackscholes, swaptions, fluidanimate, bodytrack
Med. Imaging	rician-denoise, gaussian-deblur, registration, segmentation, compressive sensing

C. Reference Designs

To evaluate the effectiveness of RHC, we compare RHC with a traditional SRAM-based cache under the same area basis. RHC is set to 4MB, which is composed of 1MB SRAM and 3MB STT-RAM, while the SRAM-based cache is set to 2MB. This setting reflects the fact that STT-RAM is about four times denser than that of SRAM. The area of the data arrays in 4MB RHC is about 0.875X that of the 2MB SRAM-based cache.

The associativity of both the 4MB RHC and 2MB SRAM-based cache are both 16-way. This setting provides the same reconfigurability on RHC and SRAM-based cache. RHC has four SRAM ways and 12 STT-RAM ways, while the SRAM-based cache has 16 SRAM ways. Both can be reconfigured from one cache way to 16 cache ways. To evaluate the effectiveness of RHC, we compare the performance and energy of RHC with three reference points: (1) *SC*: non-reconfigurable 2MB SRAM-based cache; (2) *HC*: non-reconfigurable 4MB hybrid cache (4-way SRAM + 12-way STT-RAM); and (3) *RSC*: reconfigurable 2MB SRAM-based cache. Note that the evaluation in Section V-A and Section V-C utilizes the scheme introduced in Section III-C2 for both RHC and RSC while RSC uses a single potential hit counter.

V. RESULTS

A. Effectiveness of RHC

Figure 4 shows the comparison results of L2 cache miss rate. Compared to the baseline SC, HC consistently has a lower miss rate (39% on average) because of the 2X larger cache capacity provided by the STT-RAM. But this consistent miss rate improvment is realized at the cost of more energy consumption compared to RSC and RHC, which will be discussed later. On the other hand, since RSC dynamically powers off the cache ways - although this is done based on the cache pressure - the reduced cache capacity consistently impairs the cache performance (77% more cache misses), especially when the dynamic reconfiguration scheme can not accurately capture the cache behavior. This can be observed in *bzip2* and *libquantum*. Compared to the baseline SC, there are two main scenarios with RHC: 1) for the cases where the applications have relatively large working set (such as *bzip2*, *deblur* and *compressive sensing*), RHC can achieve a considerable miss rate reduction of 52% on average; 2) for the cases where the applications have relatively small working set which can be held for a 2MB L2 cache, RHC will gradually power off half of the cache capacity. But during this process, some of the cache blocks with long reuse distance will be evicted which results in slightly higher miss rate. Overall, RHC incurs 33% more cache misses compared to SC.

Figure 5 shows the comparison results of system performance in terms of runtime of the application on the system. These results are normalized to that of the baseline SC scheme. The runtime difference of the four design schemes mainly comes from the difference of the L2 cache miss rate. Compared to the baseline SC, HC consistently has better performance (0% to 36% less runtime) because of its consistently smaller miss rate, while RSC consistently has worse performance (0% to 9% more runtime) due to its consistently larger miss rate. For the cases where RHC can achieve considerable L2 miss rate reduction, it also improves the performance (1% to 34% less runtime) over SC. For the other cases, RHC incurs a slight performance overhead (0% to 4% worse runtime).

When it comes to energy, the power of the dynamic reconfiguration begins to show gain. Figure 6 shows the comparison of memory subsystem energy, The energy data is broken down into the L1 cache dynamic/leakage energy, and the L2 cache SRAM/STT-RAM dynamic/leakage energy for detailed illustration of the energy distribution. These results are normalized to that of the baseline SC scheme. The SRAM leakage dominates the memory subsystem energy in 32nm technology. Compared to the baseline SC, HC reduces energy by 24% to 53% (30% on average), because the STT-RAM array consumes less leakage. In addition, HC consistently reduces the runtime, which reduces the SRAM leakage. By dynamically powering off the cache ways based on cache pressure, RSC can also reduce the energy by 7% to 88% (51% on average). In cases of *bzip2*, *deblur* and *compressive sensing* where the powering-on time of the remaining cache ways incurs a energy overhead which almost catches up with

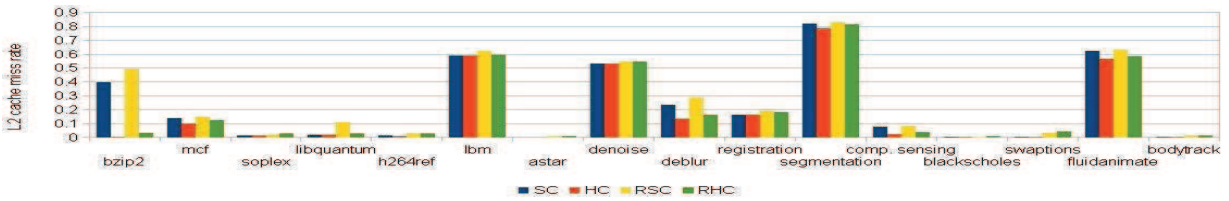


Fig. 4. Comparison results of L2 cache miss rate

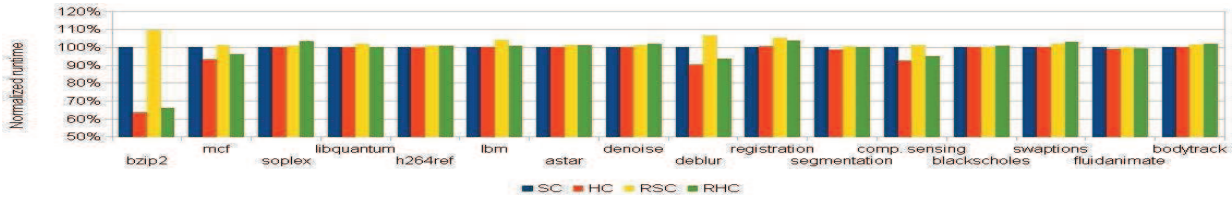


Fig. 5. Comparison results of runtime

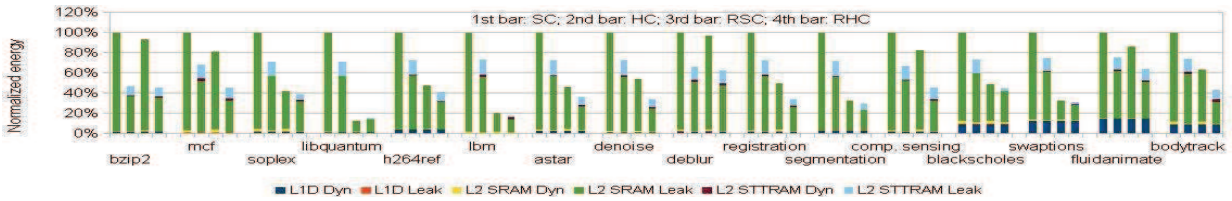


Fig. 6. Comparison results of memory subsystem energy

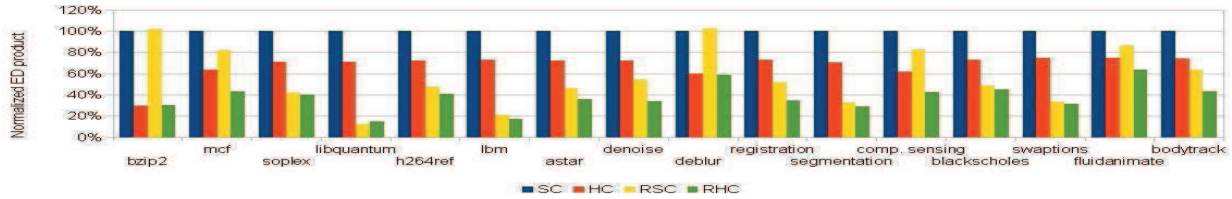


Fig. 7. Comparison results of ED product

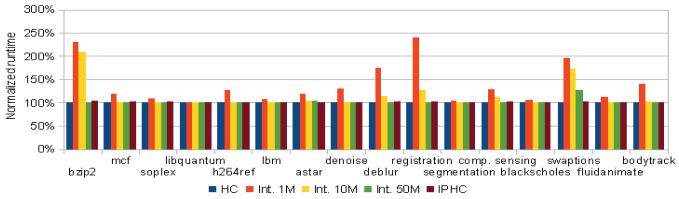
the reduction of the leakage in the powering-off cache ways, the energy reduction of RSC is much smaller than the other cases. By dynamically powering off the cache ways and maintaining the system performance, RHC achieves the least energy among all the design schemes since RHC inherits the advantages of both the low leakage NVM array and dynamic reconfiguration to save leakage. It reduces energy by 63%, 48%, 25% compared to baseline SC, HC and RSC, respectively.

To better illustrate the gain over other design schemes in terms of both energy and runtime, we use the metrics of energy-delay product (ED) to make the comparison, where the delay means the runtime. Figure 7 shows the comparison results of this metric over the four design schemes. All results are normalized to that of the baseline SC. The proposed RHC achieves the best ED among all the design schemes. On average, RHC improves the ED by 64%, 46%, and 28% compared to SC, HC and RSC, respectively.

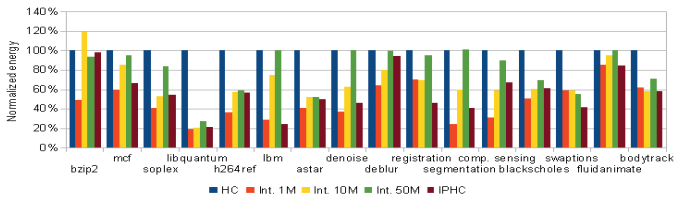
B. Comparison of Two Dynamic Schemes

For the way-based decay scheme with *independent potential hit counters* (IPHC), we evaluate three different decay intervals (1M, 10M, and 50M cycles). When the decay interval is larger, more cache ways are powered-on to maintain the performance. The largest interval we used is 50M cycles since the simulation results remains the same even when we enlarge the decay interval. Figure 8(a) shows the comparison results of runtime. The results are normalized to the baseline HC. The most critical disadvantage of the way-based decay scheme comes from the significant performance degradation, as shown in Figure 8(a). When the decay interval is set to 1M cycles, the performance drops from 2% to 131% compared to HC. Even when the decay interval is set to 50M cycles, *swaptions* still suffers from 27% performance degradation. In contrast, IPHC can provide stable performance within a 4% degradation compared to HC among all workloads.

Figure 8(b) shows the comparison of energy. IPHC can achieve better or at least similar energy reduction compared to the cases of 10M and 50M decay intervals. Therefore, IPHC can further provide energy savings when maintaining similar performance compared to the 50M decay interval case. When the decay interval is set to 1M cycles, the way-based decay scheme achieves much better energy saving on *bzip*2, *segmentation*, and *comp.sensing*. However, *bzip*2 and *comp.sensing* suffer from 122% and 24.2% performance overhead compared to IPHC. In summary, IPHC provides consistent performance compared to baseline HC while providing considerable energy saving. The way-based decay scheme suffers from potential performance degradation problem and the choice of a suitable decay interval varies from workloads.



(a) Runtime



(b) Energy consumption

Fig. 8. Comparison of runtime and energy on two dynamic schemes

C. Endurance Analysis

Table VI shows the endurance comparison between HC and RHC. The lifetime calculation is based on the method from [8] and the write cycles is 4×10^{12} [1]. RHC can achieve from 1.08X to 3.53X lifetime enhancement on most of the workloads except *bzip2*, *soplex*, *segmentation*, and *swaptions*. For *bzip2* and *segmentation*, the lifetimes of our scheme are still in a reasonable range. Through the random selection of powered-on/off victims, some write-intensive data blocks may have chance to be swapped out to main memory and thus the pressures of the most write-intensive cache blocks can be alleviated. Therefore, our reconfigurable scheme can achieve reasonable lifetime compared to HC even when available cache ways are decreased due to reconfiguration. However, we observe the non-uniform distribution of write accesses as mentioned in the previous work [1][8]. A suitable wear-leveling technique is still required to achieve better endurance.

TABLE VI
ENDURANCE COMPARISON OF 4MB NON-RECONFIGURABLE HYBRID CACHE (HC) AND 4MB RHC (UNIT: YEAR)

Workloads	HC	RHC	Workloads	HC	RHC
bzip2	299.92	200.24	g.-deblure	76.68	116.36
mcf	8.40	29.68	registration	12.88	30.68
soplex	4.64	4.6	segmentation	537.32	256.4
libquantum	2.82	4.2	comp. sensing	3.28	3.56
h264ref	22.76	41.88	blackscholes	3.144	5.44
lbm	228.96	253	swaptions	7.36	3.4
astar	16.00	30.08	fluidanimate	107.76	118.56
r.-denoise	53.44	118.8	bodytrack	9.76	10.28

VI. Conclusions

We propose an energy-efficient last-level cache design – reconfigurable hybrid cache (RHC). In RHC different memory technologies (SRAM and NVM) are unified at the same cache level to form a hybrid design, and power gating circuitry is introduced to allow adaptive powering on/off of SRAM/NVM sub-arrays at way level. Experimental results show that, the proposed the proposed RHC achieves an average 63%, 48% and 25% energy saving over non-reconfigurable SRAM-based cache, non-reconfigurable hybrid cache, and reconfigurable SRAM-based cache, while maintaining the system performance (at most 4% performance overhead).

VII. Acknowledgements

This work is partially supported by the SRC Contract 2009-TJ-1984, and the Center for Domain Specific Computing (NSF Expedition in Computing Award CCF-0926127).

References

[1] Y. Chen, W.-F. Wong, H. Li, and C.-K. Koh, "Processor caches built using multi-level spin-transfer torque ram cells," in *Proc. ISLPED 2011*, 2011, pp. 73–78.
[2] X. Wu, J. Li, L. Zhang, E. Speight, R. Rajamony, and Y. Xie, "Hybrid Cache Architecture with Disparate Memory Technologies," in *Proc. ISCA*, 2009, pp. 34–45.
[3] G. Sun, X. Dong, Y. Xie, J. Li, and Y. Chen, "A Novel Architecture of the 3D Stacked MRAM L2 Cache for CMPs," in *Proc. HPCA*, 2008, pp. 239–249.
[4] X. Dong, X. Wu, G. Sun, Y. Xie, H. Li, and Y. Chen, "Circuit and Microarchitecture Evaluation of 3D Stacking Magnetic RAM (MRAM) as a Universal Memory Replacement," in *Proc. DAC*, 2008, pp. 554–559.
[5] M. Rasquinha, D. Choudhary, S. Chatterjee, S. Mukhopadhyay, and S. Yalamanchili, "An Energy Efficient Cache Design Using Spin Torque Transfer (STT) RAM," in *Proc. ISLPED*, 2010, pp. 389–394.
[6] B. Lee, E. Ipek, O. Mutlu, and D. Burger, "Architecting Phase Change Memory as a Scalable DRAM Alternative," in *Proc. ISCA*, 2009, pp. 2–13.
[7] *International Technology Roadmap for Semiconductors (ITRS) Website*, http://www.itrs.net/Links/2009ITRS/Home2009.htm.
[8] A. Jadidi, M. Arjomand, and H. Sarbazi-Azad, "High-endurance and performance-efficient design of hybrid cache architectures through adaptive line replacement," in *Proc. ISLPED*, 2011, pp. 79–84.
[9] A. Bui, K.Cheng, J. Cong, L. Vese, Y. Wang, B. Yuan, and Y. Zou, "Platform Characterization for Domain-Specific Computing," in *Proc. ASPDAC*, 2012.
[10] C. Bienia, S. Kumar, J. P. Singh, and K. Li, "The PARSEC Benchmark Suite: Characterization and Architectural Implications," in *Proc. PACT*, 2008, pp. 72–81.
[11] P. Ranganathan, S. Adve, and N. P. Jouppi, "Reconfigurable Caches and their Application to Media Processing," in *Proc. ISCA*, 2000, pp. 214–224.
[12] M. Powell, S. Yang, B. Falsafi, K. Roy, and T. N. Vijaykumar, "Gated-Vdd: A Circuit Technique to Reduce Leakage in Deep-Submicron Cache Memories," in *Proc. ISLPED*, 2000, pp. 90–95.
[13] M. Zhang and K. Asanovic, "Fine-grain CAM-tag cache resizing using miss tags," in *Proc. ISLPED*, 2002, pp. 130–135.
[14] H. Zhou, M. C. Toburen, E. Rotenberg, and T. M. Conte, "Adaptive mode control: A static-power-efficient cache design," *ACM Trans. Embed. Comput. Syst.*, pp. 347–372.
[15] S. Kaxiras, Z. Hu, and M. Martonosi, "Cache Decay: Exploiting Generational Behavior to Reduce Cache Leakage Power," in *Proc. ISCA*, 2001, pp. 240–251.
[16] J. Chang, M. Huang, J. Shoemaker, J. Benoit, S.-L. Chen, W. Chen, S. Chiu, R. Ganesan, G. Leong, V. Lukka, S. Rusu, and D. Srivastava, "The 65-nm 16-MB Shared On-Die L3 Cache for the Dual-Core Intel Xeon Processor 7100 Series," in *JSSC*, 2007, pp. 846–852.
[17] D. Chiou, P. Jain, L. Rudolph, and S. Devadas, "Application-specific Memory Management for Embedded Systems Using Software-Controlled Caches," in *Proc. DAC*, 2000, pp. 416–419.
[18] J. Cong, K. Gururaj, H. Hunag, C. Liu, G. Reinman, and Y. Zou, "An Energy-Efficient Adaptive Hybrid Cache," in *Proc. ISLPED*, 2011, pp. 67–72.
[19] M. Qureshi, M. Franceschini, L. A. Lastras-Montaño, and J. Karidis, "Morphable Memory System: A Robust Architecture for Exploiting Multi-Level Phase Change Memories," in *Proc. ISCA*, 2010, pp. 153–162.
[20] K. Tsuchida and et al., "A 64Mb MRAM with Clamped-Reference and Adequate-Reference Schemes," in *Proc. ISSCC*, 2010, pp. 258–259.
[21] D. H. Albonesi, "Selective Cache Ways: On-Demand Cache Resource Allocation," in *Proc. MICRO*, 1999, pp. 248–259.
[22] K. Shi and D. Howard, "Challenges in Sleep Transistor Design and Implementation in Low-Power Designs," in *Proc. DAC*, 2006, pp. 113–116.
[23] P. Magnusson, M. Christensson, J. Eskilson, D. Forsgren, G. Hallberg, J. Hogberg, F. Larsson, A. Moestedt, and B. Werner, "Simics: A Full System Simulation Platform," in *IEEE Computer*, 2002, pp. 50–58.
[24] M. Martin, D. Sorin, B. Beckmann, M. Marty, M. Xu, A. Alameldeen, K. Moore, M. Hill, and D. Wood, "Multifacet's General Execution-Driven Multiprocessor Simulator(GEMS) Toolset," in *Computer Architecture News*, 2005, pp. 92–99.
[25] *CACTI 6.5*, http://www.hpl.hp.com/research/cacti/.
[26] *SPEC Benchmark*, http://www.spec.org/cpu2006, 2006.